A FRAGMENTATION REGION-BASED SKYLINE COMPUTATION FRAMEWORK FOR A GROUP OF USERS

Ghoncheh Babanejad Dehaki[1], Hamidah Ibrahim[1],
Nur Izura Udzir[1], Fatimah Sidi[1] and Ali Amer Alwan[2]

[1]Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia, Selangor, Malaysia
[2]Kulliyah of Information and Communication Technology,
International Islamic University Malaysia, Kuala Lumpur, Malaysia

ABSTRACT

Skyline processing, an established preference evaluation technique, aims at discovering the best, most preferred objects, i.e. those that are not dominated by other objects, in satisfying the user's preferences. In today's society, due to the advancement of technology, ad-hoc meetings or impromptu gathering are becoming more and more common. Deciding on a suitable meeting point (object)for a group of people (users) to meet is not a straightforward task especially when these users are located at different places with distinct preferences. A place which is close by to the users might not provide the facilities/services that meet all the users' preferences; while a place having the facilities/services that meet most of the users' preferences might be too distant from these users. Although the skyline operator can be utilised to filter the dominated objects among the objects that fall in the region of interest of these users, computing the skylines for various groups of users in similar region would mean rescanning the objects of the region and repeating the process of pair wise comparisons among the objects which are undoubtedly un-wise. On this account, this study presents a region-based skyline computation framework which attempts to resolve the above issues by fragmenting the search region of a group of users and utilising the past computed skyline results of the fragments. The skylines, which are the objects recommended to be visited by a group of users, are derived by analysing both the locations of the users, i.e. spatial attributes, as well as the spatial and non-spatial attributes of the objects. Several experiments have been conducted and the results show that our proposed framework outperforms the previous works with respect to CPU time.

KEYWORDS

Skyline Queries, Preference Queries, Group Preferences, Fragmentation Strategy.

1. INTRODUCTION

The skyline operator introduced by [6] which is used to filter a set of interesting objects from a potentially large multi-dimensional set of objects by keeping only those objects that are not worse than any other; has been greatly explored in several studies in an attempt to accurately and efficiently solve problems of real-world applications that are related to decision support and decision making. It attempts to derive the best, most preferred set of objects known as skylines according to a set of established criteria. The process of computing skylines becomes more challenging when conflicting criteria are involved while the number of criteria to be considered is huge. A classic example is selecting a hotel for a holiday whereby hotels that are close to the beach are

David C. Wyld et al. (Eds): AIMLA, DBDM, CCNET - 2021
pp. 35-50, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.111303

known to be expensive. While other criteria like facilities, rating, service, etc are equally important, distance and price are examples of conflicting criteria.Although there are a considerable amount of works in processing skylines, however most of them are limited to the aim of satisfying the preferences of a single user [5, 6, 8, 11, 15].Today's advancement of technology shows that ad-hoc meetings or unplanned gathering are becoming more and more common. Determining the best, most preferred objects in which the preferences of several users are to be considered is more complex as opposed to a single user. The following scenario simulates a sample of situation considered in this paper.

Assume a group of users who are located at different locations; would like to gather and hence have to decide on a place to meet. Several criteria need to be considered such as the location of the place, i.e. how far it is from the location of each user (spatial attribute), the opening hour, food, ticket price, rating, facilities provided, etc (non-spatial attributes). Intuitively, deciding on the best meeting point for these users is not a straightforward task as many criteria need to be considered. A place which is near to the users might not be a place that meets all the users' preferences. While a place which provides facilities/services that meet most of the users' preferences might be located far away from these users. Thus, it is essential to have a method that could find an object(s) within a predetermined region that dominates other objects with respect to both the spatial (location) and non-spatial attributes of the objects that best suits the preferences of a group of users. Although this has been well tackled by [9, 19, 26, 27], there is no attempt made to resolve the following issue. Obviously, similar regions may have to be explored again in computing skylines for different groups of users (or even the same group of users at a different day/time). Attempting to rescanning the objects of previously visited regions and recomputing the skylines of the regions (i.e. repeating the process of pairwise comparisons among objects) are undoubtedly unwise and costly.

Motivated by the above example, we propose a region-based skyline computation framework, *RSGU*, an enhancement of our previous framework, *SGMU* [9], with the main aim at avoiding the process of rescanning the objects of a previously visited region by utilising a fragmentation strategy as well as re computing the skylines for a group of users by utilising the past computed skyline results of the fragments. The skylines, which are the objects recommended to be visited by the group of users, are derived by analysing both the locations of these users, i.e. spatial attributes, as well as the spatial and non-spatial attributes of the objects.

This paper is organised as follows: Section 2 presents the related works which are organised into two parts, namely: skyline algorithms for a single user and skyline algorithms for a group of users. This is followed by Section 3 which introduces the notations and deliberates the terms that are used throughout this paper. It also presents the problem tackled by this paper. Section 4 presents our proposed framework and the steps to be performed in order to achieve the main aim of the work. Section 5 discusses the initial results achieved by our proposed framework while the last section, Section 6, gives the summary of the paper.

2. RELATED WORK

The skyline operator proposed by [6] is a well-studied technique for filtering the best, most preferred objects from a multi-dimensional set of objects. Since then many variants of skyline algorithms have been proposed, each tackling a slightly different issue mainly due to the nature of data being handled. We categorised these skyline algorithms into two main categories, namely: skyline algorithms for a single user and skyline algorithms for a group of users.

Skyline algorithms for a single user – Generally, these skyline algorithms filter the best, most preferred objects from a potentially large multi-dimensional set of objects with the assumption that all users have the same property with the same objective function (preferences). Among the earlier and most cited skyline algorithms in the literature are *Block Nested Loop* (*BNL*) [6], *Divide-and-Conquer* (*D&C*) [6], *Linear Elimination Sort for Skyline* (*LESS*) [11], *Branch and Bound Skyline* (*BBS*) [24], *SkyCube* [6], and *Sort and Limit Skyline algorithm* (SaLSa) [5]. These algorithms attempt to resolve the optimisation problem which is proven through the reduction of the processing time. Later due to advancement in technology that produces gigantic amount of various forms of data, several skyline algorithms have been proposed. These algorithms attempt not only to resolve the optimisation problem but also issues related to the uncertainty of data which is defined as the degree to which data are inaccurate, imprecise, untrusted, unknown or incomplete. These include among others *ISkyline* [14], *sorting-based bucket skyline* [18], *Incoskyline* [2], *Jincoskyline* [1], and *OIS* [12] that handle the issues of incompleteness of data; *probabilistic skyline model* [25], *τ-Skyline* [29], *SkyQUD* [20, 21, 22, 23] and *SkyQuiD* [17] focus on the challenges in computing skyline queries for uncertain database; the works by [4] and [10] attempt to solve the issues related to uncertain data in a data stream; while the work by [3] focuses on dynamic database. Nonetheless, these algorithms are specifically designed to cater only a single user query.

Skyline algorithms for a group of users – These skyline algorithms keep those objects that are not worse than any other from a potentially large multi-dimensional set of objects in which the preferences of multiple users are taken into account. As we assume that the objects are static, hence we further elaborate only those works that are similar to our intention. To the best of our knowledge the only works that contribute to skyline queries for a group of users are the works done by [19], [26], and[27]. In processing spatial skyline query for a group of users, [26] have proposed two algorithms, namely: B^2S^2 and VS^2. The B^2S^2 algorithm utilises the *R*-tree while the VS^2 algorithm utilises the Voronoi diagram. Both algorithms are performed on static user points. Later, [27] proposed VCS^2, an enhancement to B^2S^2 and VS^2, which aims at processing skyline query by taking into consideration the movements of the users. However, VCS^2 only calculates the last location of the users and does not consider the changes of locations to prevent recalculation of the skylines. In [19], the authors proposed an algorithm, *VR* (Voronoi and *R*-tree), to find spatial skylines for a group of user points. In their work the user points and objects are considered static. The two data structures, *R*-tree and Voronoi, used in [27] are combined in this work. Both the spatial and non-spatial attributes of the objects are analysed to find the skylines. Meanwhile, our previous solution, *SGMU* [9], is designed with the main aim to continuously derive skylines for a group of mobile users. In *SGMU*, while the users decide on a place to visit, the skylines are continuously updated since a place that was initially in the top list based on the users' locations at time $t_a$might no longer be the place of interest at time t_b where $t_a < t_b$ since the users' locations at time t_a might be different at time t_b.

Although [9, 19, 26, 27] considered the spatial attributes of the group of users in determining the skylines, but there is no attempt made to avoid rescanning of objects of previously visited regions and simultaneously avoid repeating the process of pairwise comparisons among the objects.

3. PRELIMINARIES

This section elaborates the concepts that are related to the work presented in this paper. It also defines the terms and introduces the notations used throughout the paper. Towards the end of this section, we formulate the problem tackled in this paper. To clarify the concepts and steps proposed in this work, the following sample of data will be used. Table 1(a) and Table 1(b) present the spatial attribute (*Location*) of the users of group a, G_a, and group b, G_b, respectively. Here, we assume that the request submitted by G_a is at time t_a, while the request submitted by G_b is at time t_b where $t_a < t_b$. Table 2 presents the spatial (*Location*) and non-spatial (*Rate*, *Fee*) attributes of the objects. For the non-spatial attributes, we assume higher rate and lower fee are preferable.

Table 1. The spatial attribute of the users.

ID	Location
u_1	(8, 8)
u_2	(14, 16)
u_3	(2, 5)

(a) Group a, G_a

ID	Location
u_1	(5, 8)
u_2	(10, 10)
u_3	(18, 10)

(b) Group b, G_b

Table 2. The spatial and non-spatial attributes of the objects.

Restaurant	Location	Rate	Price	Restaurant	Location	Rate	Price
o_1	(2, 3)	3	70	o_{24}	(4, 13.3)	3	75
o_2	(3, 4)	4	65	o_{25}	(7, 13)	1	90
o_3	(3, 1)	5	80	o_{26}	(16, 15)	2	86
o_4	(7, 1.7)	2	75	o_{27}	(20, 14)	5	80
o_5	(6, 5)	3	65	o_{28}	(23, 20)	3	60
o_6	(7, 7)	5	70	o_{29}	(21, 21)	5	62
o_7	(9, 8)	1	80	o_{30}	(17, 23)	4	95
o_8	(8, 9.7)	2	85	o_{31}	(14, 20)	2	65
o_9	(7, 11)	4	73	o_{32}	(13, 18)	2	55
o_{10}	(10, 5)	3	50	o_{33}	(10, 19)	3	70
o_{11}	(10.7, 6)	1	65	o_{34}	(1, 16)	4	62
o_{12}	(15, 2)	2	80	o_{35}	(3, 22)	4	81
o_{13}	(17, 1)	5	105	o_{36}	(7, 20)	3	90
o_{14}	(22, 4.7)	4	90	o_{37}	(24, 15)	2	66
o_{15}	(17, 5.7)	3	85	o_{38}	(-3, -1)	1	57
o_{16}	(20, 7)	4	90	o_{39}	(-1, 7)	1	61
o_{17}	(23, 9)	1	55	o_{40}	(10.3,13)	4	71
o_{18}	(16, 8)	2	54	o_{41}	(-4, 4)	3	98
o_{19}	(14, 10)	4	80	o_{42}	(8, -2)	2	58
o_{20}	(11, 9.7)	5	56	o_{43}	(8, 18)	2	85
o_{21}	(4, 10)	3	67	o_{44}	(-2, 10)	4	70
o_{22}	(2, 12)	5	100	o_{45}	(3, -1)	5	80
o_{23}	(3, 13)	4	74				

Given a dataset $D = < A, U, O >$, where $U = \{u_1, u_2, \dots, u_n\}$ is a list of $n$users, $O = \{o_1, o_2, \dots, o_m\}$ is a list of $m$objects, and $A = < A_S, A_N >$ are the criteria (dimensions) to be considered in the skyline computation where A_S is a spatial attribute while $A_N = \{d_1, d_2, \dots, d_l\}$is a set of non-spatial attributes. Based on Table 2, $A_S = Location$ and $A_N = \{Rate, Price\}$.

The following definitions defined the properties of a user and an object as used in ourwork.

Definition 1 *Property of a User*: Each user, $u_i \in U$, is associated with a spatial attribute which represents the location of the user at time, t. This is denoted by $u_i(x_i, y_i)$ where x_i and y_i represent the latitude and longitude coordinates, respectively. For instance, $u_1(8, 8)$ of Table 1(a) denotes the location of user u_1 where $x_i = 8$ and $y_i = 8$.

Definition 2 *Properties of an Object*: Each object $o_j \in O$ has two main elements denoted by $o_j = (s_j, ns_j)$ where s_j is the value of spatial attribute (location), A_S, and

$ns_j = \{o_j.d_1, o_j.d_2, \ldots, o_j.d_l\}$ is a set of values of non-spatial attributes, A_N, associated to o_j. The location of an object $o_j \in O$ is denoted by $o_j(x_j, y_j)$. As we assume that each object $o_j \in O$is static, thus the location of the object is fixed regardless the changes in time. Hence, $o_j = (s_j, ns_j)$ can be written as $o_j = ((x_j, y_j), \{o_j.d_1, o_j.d_2, \ldots, o_j.d_l\})$. For instance, the object $o_1$of Table 2 can be written as $o_1 = ((2,3), \{3,70\})$.

The following definitions defined the notion of dominance in ourwork.

Definition 3 *Dominance*: Given two objects $o_i = (s_i, ns_i) \in O$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$, o_i is said to dominate o_j (denoted by $o_i \prec o_j$) if and only if both of the following conditions hold:

(1) o_i non-spatially dominates $o_j (o_i \prec_{ns} o_j)$ and
(2) o_i spatially dominates $o_j (o_i \prec_s o_j)$.

Without loss of generality, this definition is applicable for a given bounded space, S, i.e. O is a set of objects in the space S. Similar note applies for *Definition* 4 and *Definition* 5.

Definition 4*Non-spatial Dominance*: Given two objects $o_i = (s_i, ns_i) \in O$and $o_j = (s_j, ns_j) \in O$ where $i \neq j$, o_i is said to non-spatially dominate o_j(denoted by $o_i \prec_{ns} o_j$) if and only if $o_i$is no worse than (in this definition, greater value is preferable) o_j in all the non-spatial attributes, A_N. This is formally written as follows: $o_i \prec_{ns} o_j$ if and only if $\forall d_k \in A_N, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in A_N, o_i.d_l > o_j.d_l$. For instance, given $o_6 = ((7,7), \{5,70\})$ and$o_{12} = ((15,2), \{2,80\})$, $o_6 \prec_{ns} o_{12}$ since o_6 is better than o_{12} in both the dimensions *Rate* and *Price*.

Definition 5 *Spatial Dominance*: Given two objects $o_i = (s_i, ns_i) \in O$ and $o_j = (s_j, ns_j) \in O$ where $i \neq j$, o_i is said to spatially dominate o_j (denoted by $o_i \prec_s o_j$) if and only if for every user $u_k \in U$, the distance between o_i and $u_k$is no worse than the distance between $o_j$and u_k. This is formally written as follows: $o_i \prec_s o_j$ if and only if $\forall u_k \in U, dist(o_i, u_k) \leq dist(o_j, u_k) \wedge \exists u_l \in U, dist(o_i, u_l) < dist(o_j, u_l)$. For instance, the distances between $o_1 = ((2,3), \{3,70\})$ and u_1, u_2, and u_3 of group G_a are 7.81, 17.69, and 2, respectively; while the distances between $o_2 = ((3,4), \{4,65\})$ and u_1, u_2, and u_3 of group G_a are 6.4, 16.27, and 1.41, respectively. Thus, $o_2 \prec_s o_1$.

Definition 6 is an extension of *Definition* 3 in which the dominance testing is performed over a predefined space. The list of objects considered in *Definition* 3 is the $m$objects as defined in the system while in *Definition* 6 the list of objects is confined to those objects that fall within a certain space.

Definition 6 *Dominance in a Space*: Given a bounded space, S (region, MBR, fragment, area, polygon, etc), and two objects $o_i = (s_i, ns_i) \in O$and $o_j = (s_j, ns_j) \in O$ where $i \neq j$ in S, o_i is said to dominate o_j (denoted by $o_i \prec o_j$) in S if and only if

(1) o_i non-spatially dominates $o_j (o_i \prec_{ns} o_j)$ in S and
(2) o_i spatially dominates $o_j (o_i \prec_s o_j)$ in S.

Definition 7 *Skylines of a Space*: An object $o_i \in O$ in a space S is a skyline of S if there are no other objects $o_j \in O$ in the space S where $i \neq j$ that dominates o_i. In this paper, $SkyG_p$ is used to denote the skyline set for the group $G_p$of a given space S.

Based on the above definitions, we now formulate the problem that is tackled by this paper.

Problem Formulation

Given a group of users, $G_p = \{u_1, u_2, \dots, u_p\}$, where $G_p \subset U$, and the list of candidate skylines of G_p, CS_{G_p}, in region S_{G_p}. Find the skylines of a group of users $G_q = \{u_1, u_2, \dots, u_q\}$ in region S_{G_q} where $G_q \subset U$, $G_q \neq G_p$, and$S_{G_p} \cap S_{G_q} \neq \{\}$ by utilising CS_{G_p} that has been derived for G_p. This is depicted in Figure 1 where the area covered to compute the skylines for $G_q$that falls in the region S_{G_q} can be reduced to the area defined by $S_{G_q} - S_{G_p}$, while the skyline computation that has been performed earlier over the area $S_{G_p} \cap S_{G_q}$ for $G_p$can be avoided by simply utilising the obtained results derived earlier for G_p, i.e. CS_{G_p}.

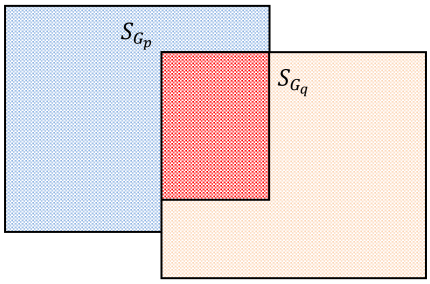


Figure 1. The reduction area in deriving skylines for a group of users

4. THE PROPOSED FRAMEWORK

This section elaborates the extended framework, *RSGU* [9], which we have proposed in order to solve the problem defined in Section 3. The framework is presented in Figure 2. It consists of seven main steps that are: (1) Identify the centroid, (2) Construct a search region, (3) Identify the overlapping region, (4) Construct the fragments of a search region, (5) Derive non-spatial skylines, (6) Derive spatial skylines, and (7) Derive the final skylines. Step (3) is conducted only when past computed skyline results of the fragments are available. Step (3) and Step (4) are the new steps incorporated into *RSGU* [9]. Each of these steps is elaborated in the following subsections.

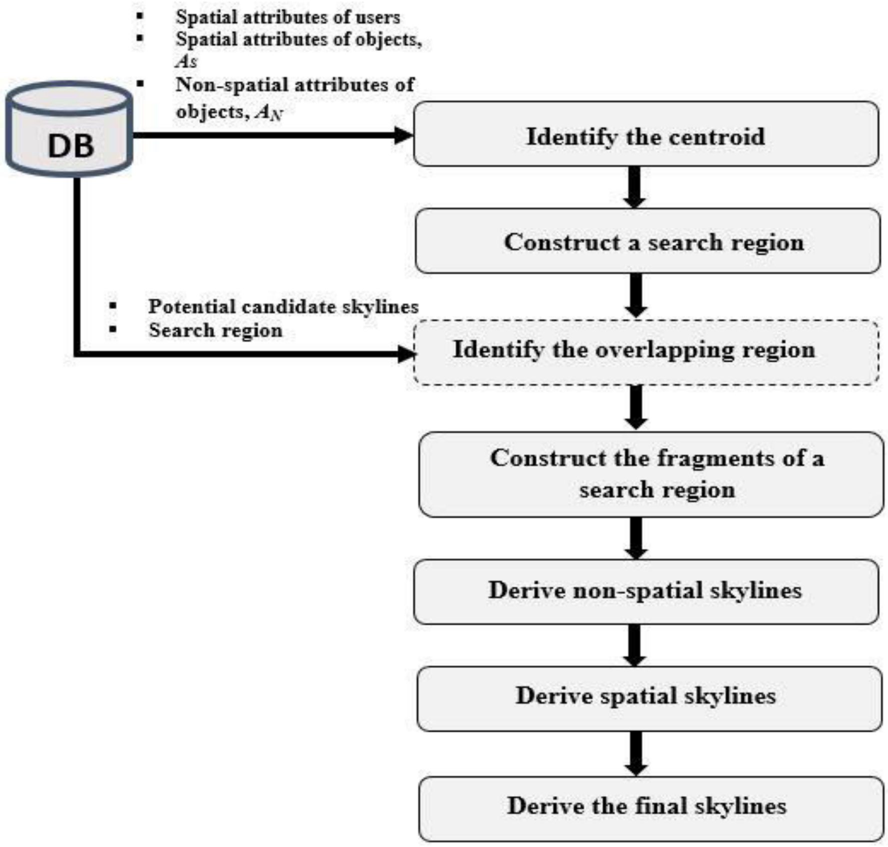


Figure 2. The proposed framework

4.1. Identify the Centroid

When a group of users, $G_p = \{u_1, u_2, \dots, u_p\}$, decided to meet, there must be a point to guide the direction of their movements. In our work, we assume that the group of users will move towards a point that has the tendency to be a center based on the users' locations. This point is called centroid and is denoted by$C_{G_p}(x_{G_p}, y_{G_p})$. The centroid of a given group of users, C_{G_p}, is determined using the following formula [13]:

$$C_{G_p}\left(x_{G_p} = \frac{\sum_{i=1}^{n} xi}{n}, y_{G_p} = \frac{\sum_{i=1}^{n} yi}{n}\right) \qquad (1)$$

where x_i is the x coordinate of user u_i location, $y_i$is the y coordinate of user u_i location, x_{G_p} is the average of the x coordinates of all users in the group G_p, and $y_{G_p}$is the average of the y coordinates of all users in the group G_p. Based on the example given in Table 1(a), the centroid of G_a is $C_{G_a}(8, 9.6)$.

4.2. Construct a Search Region

The aim of constructing a search region is to limit the searching space to those spaces in which potential candidate skylines (objects) are derived. Since we are interested with a group of users, thus the searching space should include the regions of interest of all users in the group. This is achieved by: (1) identifying the search region for each user, S_{u_i} and (ii) identifying the search region given a group of users, S_{G_p}.

4.2.1. Identify the search region for each user, S_{u_i}

Since the centroid of a given group of users, say C_{G_p}, which is identified in the previous step does not necessarily contain an object, therefore the nearest object, o_n, to the centroid C_{G_p}will have to be determined. The nearest object is an object with the shortest Euclidean distance from the centroid, i.e. $\{o_n | o_n \in O \wedge \forall o_i \in O - \{o_n\}: Ed(C_{G_p}, o_n) < Ed(C_{G_p}, o_i)\}$ where $Ed$is the Euclidean distance function. Based on the example given in Table 1(a), the nearest object to the centroid of G_a, i.e. $C_{G_a}(8, 9.6)$, is $o_8(8, 9.7)$. The search region for a user, u_i, denoted as S_{u_i}, is the area bounded by a rectangle also known as the minimum bounding rectangle, MBR_{u_i}.We use the notation S_{u_i} to denote the search region of u_i while MBR_{u_i} is used in forming the S_{u_i}. The distance between a user, u_i, and the nearest object, o_n, denoted by $R_{u_i o_n}$, is calculated by the following equation:

$$R_{u_i o_n} = \sqrt{(x_{o_n} - x_i)^2 + (y_{o_n} - y_i)^2} \quad (2)$$

where x_i is the x coordinate of user u_i location, y_i is the y coordinate of user u_i location, x_{o_n} is the x coordinate of object o_n location, and y_{o_n} is the $y$coordinate of object o_n location. AMBR is formed based on four vertices as explained in the following: the vertex at the bottom left of the MBR is denoted by $bl = (x_{bl}, y_{bl})$; the vertex at the bottom right of the MBR is denoted by $br = (x_{br}, y_{br})$; the vertex at the top left of the MBR is denoted by $tl = (x_{tl}, y_{tl})$; and the vertex at the top right of the MBR is denoted by$tr = (x_{tr}, y_{tr})$. Figure 3 depicts these notations. These vertices are calculated as follows:

$$bl = (x_i - R_{u_i o_n}, y_i - R_{u_i o_n})$$
$$br = (x_i + R_{u_i o_n}, y_i - R_{u_i o_n})$$
$$tl = (x_i - R_{u_i o_n}, y_i + R_{u_i o_n})$$
$$tr = \left(x_i + R_{u_i o_n}, y_i + R_{u_i o_n}\right)$$

4.2.2. Identify the search region given a group of users, S_{G_p}

This step is simply achieved by performing union on the search region of each user in the group, i.e. $S_{G_p} = \bigcup_{i=1}^{p} S_{u_i}$. An example of a search region $S_{G_a} = \bigcup_{i=1}^{3} S_{u_i}$ can be seen in Figure 4.

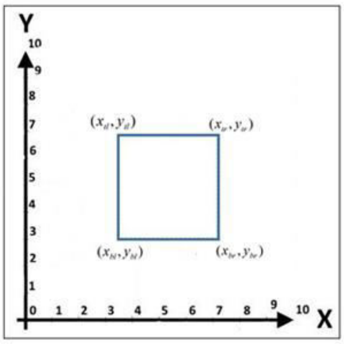


Figure 3. Minimum Bounding Rectangle (MBR)

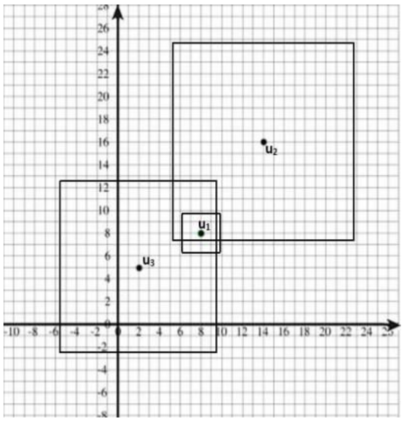


Figure4. The search region for a group of users

4.3. Construct the Fragments of a Search Region

This step partitions the search region of a group of users, S_{G_p}, into m fragments (subspaces). Here, the vertices of the MBRassociated to each $S_{u_i}$are analysed and sorted according to the x- and y-axes.The search region (space) is vertically fragmented based on the x-coordinates, while it is horizontally fragmented based on the y-coordinates. The$MRBs$ formed within the $S_{G_p}$are the fragments of the region.

Objects that fall within each fragment are then identified. Given an object, $o_j(x_j, y_j)$, and a fragment, F_i, with $bl(x_l, y_b)$, $br(x_r, y_b)$, $tl(x_l, y_t)$, and $tr(x_r, y_t)$, the following cases are identified:

(a) If $x_l < x_j < x_r$ and $y_b < y_j < y_t$, then the object $o_j(x_j, y_j)$ is said to fall within the boundary of fragment, F_i.

(b) If $x_j = x_l$or $x_j = x_r$or $y_j = y_b$ or $y_j = y_t$, then the object $o_j(x_j, y_j)$ is said to intersect with the boundary of fragment, F_i.

(c) Objects that do not meet the above two cases are objects that are outside the boundary of fragment, F_i.

Further, utilising the non-spatial dominance testing given in *Definition* 8, an extension to the *Definition 4*,over the objects that satisfy the cases (a) or (b) above, denoted by O_{F_i}, the non-spatial candidate skylines of a fragment are determined, $CS_{ns_{F_i}}$, as defined by *Definition* 9.

Definition 8 *Non-spatial Dominance of the Fragment F_i*: Given two objects $o_i = (s_i, ns_i) \in O_{F_i}$and $o_j = (s_j, ns_j) \in O_{F_i}$where $i \neq j$, $o_i$is said to non-spatially dominate o_j(denoted by $o_i \prec_{ns} o_j$) if and only if $o_i$is no worse than (in this definition, greater value is preferable) $o_j$in all the non-spatial attributes, A_N. This is formally written as follows: $o_i \prec_{ns} o_j$ if and only if $\forall d_k \in A_N, o_i.d_k \geq o_j.d_k \wedge \exists d_l \in A_N, o_i.d_l > o_j.d_l$.

Definition 9 *Candidate Skylines of the Fragment F_i*: An object $o_i \in O_{F_i}$ in a space $F_i$is a non-spatialcandidate skyline of F_i if there are no other objects $o_j \in O_{F_i}$ in the space $F_i$where $i \neq j$ that non-spatially dominates o_i.

This will avoid rescanning the objects of the region and repeating the process of pairwise comparisons among the objects during the computation of subsequent skyline queries. Figure5 presents the fragments constructed based on the S_{G_a} given in Figure4. The x-coordinates = {-5.6, 0, 5.3, 6.3, 9.6, 9.7, 22.7} and y-coordinates = {-2.6, 0, 6.3, 7.3, 9.7, 12.6, 24.7}. Altogether there are 28 fragments; some samples are given in Table 3.

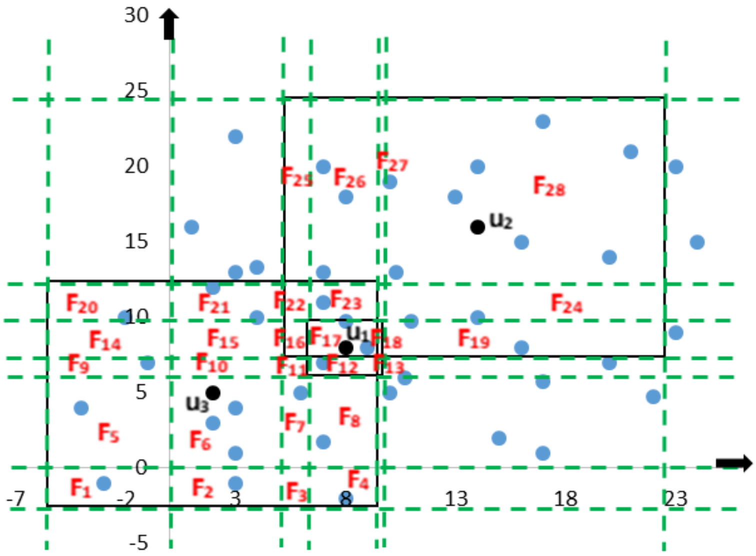


Figure5. The fragments derived based on the S_{G_a} given in Figure4

Table 3. Sample of fragments and their associated candidate skylines

x Coordinate (x_l, x_r)	y Coordinate (y_b, y_t)	$bl(x_l, y_b)$	$br(x_r, y_b)$	$tl(x_l, y_t)$	$tr(x_r, y_t)$	Fragment, F_i	Objects	Candidate skylines, CS_{nsF_i}
-5.6, 0	-2.6, 0	-5.6, -2.6	0, -2.6	-5.6, 0	0, 0	F_1	o_{38}	o_{38}
0, 5.3	-2.6, 0	0, -2.6	5.3, -2.6	0, 0	5.3, 0	F_2	o_{45}	o_{45}
5.3, 6.3	-2.6, 0	5.3, -2.6	6.3, -2.6	5.3, 0	6.3, 0	F_3	-	-
6.3, 9.6	-2.6, 0	6.3, -2.6	9.6, -2.6	6.3, 0	9.6, 0	F_4	o_{42}	o_{42}
…	…	…	…	…	…	…	…	…
0, 5.3	0, 6.3	0, 0	5.3, 0	0, 6.3	5.3, 6.3	F_6	o_1, o_2, o_3	o_2, o_3
…	…	…	…	…	…	…	…	…
9.7, 22.7	7.3, 24.7	9.7, 7.3	22.7, 7.3	9.7,24.7	22.7, 24.7	F_{28}	$o_{26}, o_{27}, o_{29}, o_{30}, o_{31}, o_{32}, o_{33}, o_{40}$	o_{29}, o_{32}

4.4. Derive Non-Spatial Skylines

This step performs the non-spatial dominance testing given in *Definition* 8 towards the CS_{nsF_i} lists derived in the previous step to generate the non-spatial skylines of a given group of users. In other words, the pair wise comparisons are only performed between objects that are the candidate skylines of a fragment. The objects that non-spatially dominate the other objects, given the CS_{nsF_i} lists where $i = \{1, 2, \ldots, 28\}$ in Table 3 are o_{18} and o_{20}, thus $Sky_{nsG_a} = \{o_{18}, o_{20}\}$.

4.5. Derive Spatial Skylines

This step applies the spatial dominance testing given in *Definition* 5 towards the $CS_{ns_{F_i}}$lists. It calculates the distance between each object and each user as well as the sum of the distances (*Sum Distance*). The sequence of comparisons between these objects is based on the lowest value of *Sum Distance*. An example is shown in Table 4; $o_8$will be the first object selected which is then followed by o_7. The objects that spatially dominate the other objects, given the $CS_{ns_{F_i}}$listsin Table 3 are as listed in $Sky_{s_{G_a}} = \{o_2, o_5, o_6, o_7, o_8, o_9, o_{20}, o_{25}, o_{26}, o_{32}, o_{40}\}$.

Table 4. The distance and sum distance of each object in $CS_{ns_{F_i}}$

Restaurant	u_1	u_2	u_3	Sum Distance	Restaurant	u_1	u_2	u_3	Sum Distance
o_1	7.81	17.69	2	27.5	o_{27}	13.41	6.32	20.12	39.85
o_2	6.4	16.27	1.41	24.08	o_{29}	18.38	8.6	24.83	51.81
o_3	8.6	18.6	4.12	31.32	o_{30}	18.6	7.61	23.43	49.64
o_4	6.37	15.92	5.99	28.28	o_{31}	13.41	4	19.2	36.61
o_5	3.6	13.6	4	21.2	o_{32}	11.18	2.23	17.02	30.43
o_6	1.41	11.4	5.38	18.19	o_{33}	11.18	5	16.12	32.3
o_7	1	9.43	7.61	18.04	o_{36}	12.04	8.06	15.81	35.91
o_8	1.7	8.7	7.62	18.02	o_{38}	14.21	24.04	7.81	46.06
o_9	3.16	8.6	7.81	19.57	o_{39}	9.05	17.49	3.6	30.14
o_{18}	8	8.24	14.31	30.55	o_{40}	5.5	4.76	11.52	21.78
o_{19}	6.32	6	13	25.32	o_{41}	12.64	21.63	6.08	40.35
o_{20}	3.44	6.97	10.15	20.56	o_{42}	10	18.97	9.21	38.18
o_{21}	4.47	11.66	5.38	21.51	o_{43}	10	6.32	14.31	30.63
o_{22}	7.21	12.64	7	26.85	o_{44}	10.19	17.08	6.4	33.67
o_{25}	5.09	7.61	9.43	22.13	o_{45}	8.6	20.24	6.08	34.92
o_{26}	10.63	2.23	17.2	30.06					

4.6. Derive the Final Skylines

This is the final step that combines the results produced in the steps presented in subsections 4.4 and 4.5 above. Based on *Definition* 7, the final skylines for a given group G_i is given by, $Sky_{G_i} = Sky_{ns_{G_i}} \cup Sky_{s_{G_i}}$. Thus, the final skylines for the group G_a, $Sky_{G_a} = \{o_2, o_5, o_6, o_7, o_8, o_9, o_{18}, o_{20}, o_{25}, o_{26}, o_{32}, o_{40}\}$.

4.7. Identify the Overlapping Region

This step constructs the overlapping region, O_R, between the search regions of two groups of users, say S_{G_i} and S_{G_j}. We assume that the results of the skyline queries of a group of users, say G_i, have been derived. Thus, the overlapping region indicates that the region has been scanned and it is unwise to scan it again. Figure 6 shows two search regions, S_{G_a}, the polygon with black border line and S_{G_b}, the polygon with red border line which represent the search region of group G_a and group G_b, respectively.

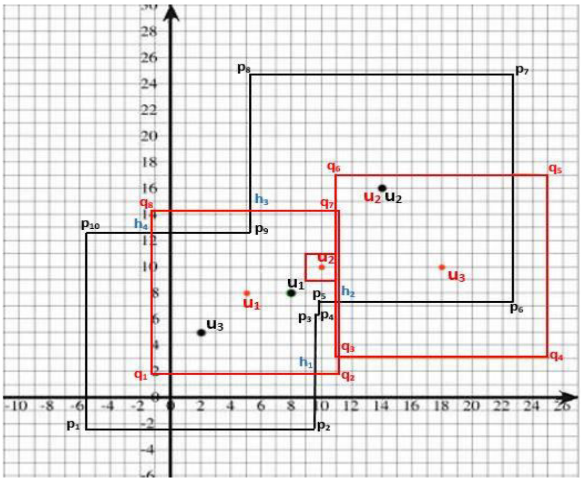


Figure6. The overlapping region between S_{G_a} and S_{G_b}

To identify the overlapping region, the following steps are performed:

(1) Get the polygon's vertices of S_{G_i}. Based on our example, $S_{G_a} = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}\}$. Note that for simplicity, we omit the coordinates of the vertices.
(2) Get the polygon's vertices of S_{G_j}. Based on our example, $S_{G_b} = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$.
(3) Get the vertices of S_{G_i}that are also in S_{G_j}. Based on our example, $I_{G_a-G_b} = \{p_3, p_4, p_5, p_6, p_9\}$.
(4) Get the vertices of S_{G_j}that are also in S_{G_i}. Based on our example, $I_{G_b-G_a} = \{q_1, q_6, q_7\}$.
(5) Get the coordinates where the edges of S_{G_i} and S_{G_j} meet. Based on our example, $H = \{h_1(9.6, 1.8), h_2(11.2, 7.3), h_3(22.7, 17), h_4(5.3, 14.2), h_5(-1.2. 12.6)\}$.
(6) The overlapping region, O_R, is defined as a polygon derived based on the following ces:$I_{G_a-G_b} \cup I_{G_b-G_a} \cup H$. Based on our example, $O_R = \{h_1, p_3, p_4, p_5, h_2, p_6, h_3, q_6, q_7, h_4, p_9, h_5, q_1\}$.

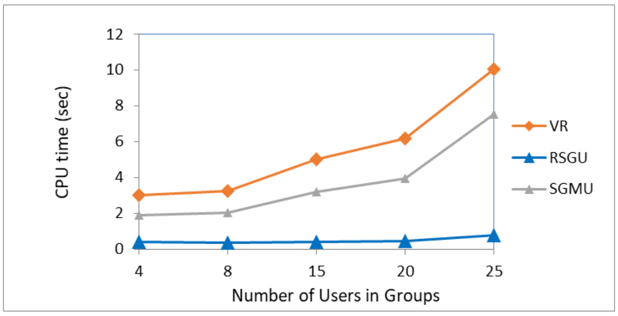
Once the O_R has been defined, the fragments derived in the earlier step are analysed. Those fragments that fall within the O_R; are retrieved together with their candidate skylines, CS_{OR}. Hence, scanning this area is no longer necessary. While for the non-overlapping area, denoted as $\neg O_R$, the following steps as discussed above will be conducted: (4) Construct the fragments of the non-overlapping region, i.e. $\neg O_R = S_{G_j} - O_R$ (5 and 6) Derive non-spatial skylines and spatial skylines, respectively by considering both the lists CS_{OR} and $CS_{\neg OR}$, and (7) Derive the final skylines.

5. PERFORMANCE EVALUATION

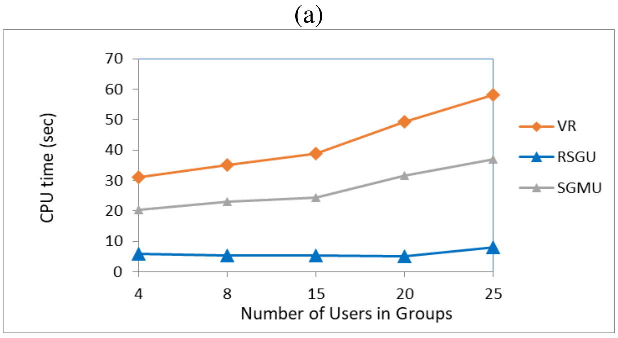
In this section, we present the initial results of the experiments that we have conducted. The experiments are conducted on a PC with Intel core™ i7 processor, 2.50 GHz CPU, 16GB main memory, and 900GB hard disk. We used both real and synthetic datasets. The real dataset is obtained from median of each road line fragment data of Long Beach from the TIGER database [28]. The dataset contains 50,747 points standardised in [0, 1000] [0, 1000] space. We used synthetic dataset consisting of 100 points with different densities standardised in [0, 1000] [0, 1000] space. The density in synthetic dataset and real dataset of TIGER database is based on the number of point's falls into one square unit in normal. We ran our proposed framework, *RSGU*, *SGMU*

[9], and *VR* algorithm [19] using randomly selected user points and objects in each dataset. Each dataset contains a spatial attribute and two non-spatial attributes. Three experiments have been conducted as elaborated below.

Effect of number of users in a group – Figure 7 presents the experimental results of *RSGU*, *SGMU* [9], and *VR* algorithm [19] for both the (a) syntactic dataset and (b) real dataset with respect to the CPU time when the number of users in a group is varied. In this experiment we varied the number of users in a group from 4 to25whilethenumberofobjectsisfixedto100and50000, respectively with 32 number of groups of users, and 50% of overlapping region. It is obvious, when the number of users in a group increases, the CPU time also increases. Nonetheless, our proposed framework, *RSGU*, outperforms both the *SGMU* and *VR* algorithm; while *SGMU* is better than *VR* algorithm.



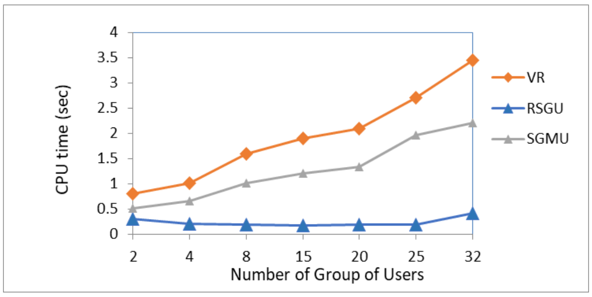
(a)



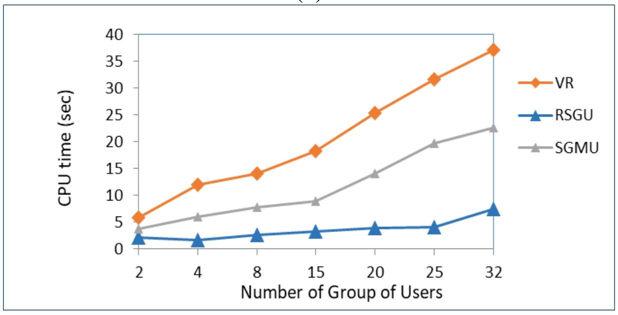
(b)

Figure 7. CPU time with varying number of users in a group over (a) synthetic dataset and (b) real dataset

Effect of number of groups – Figure 8 presents the experimental results of *RSGU*, *SGMU* [9], and *VR* algorithm [19] for both the (a) syntactic dataset and (b) real dataset with respect to the CPU time when the number of groups is varied. In this experiment, the number of users in a group is fixed to 10, the number of objects is fixed to100and50000, respectively with 50% of overlapping region while the number of groups of users is varied from 2 to 32. It is obvious that when the number of groups increases, the CPU time also increases. Nevertheless, our proposed framework, *RSGU*, outperforms both the *SGMU* and *VR* algorithm; while *SGMU* is better than *VR* algorithm.



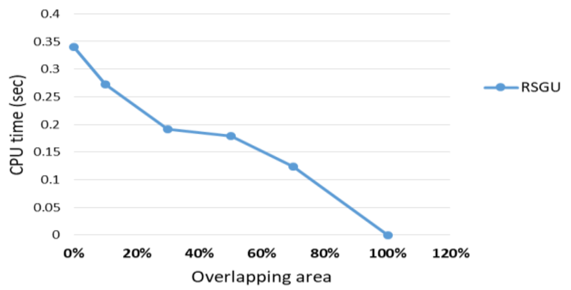
(a)



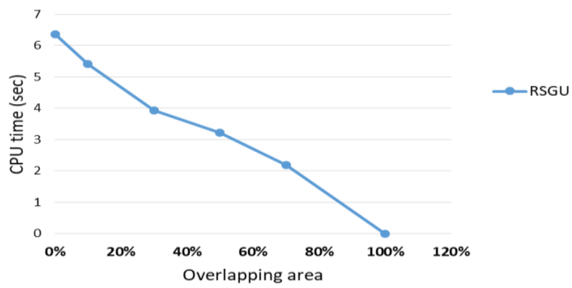
(b)

Figure8. CPU time with varying number of groups of users over (a) synthetic dataset (b) real database

Effect of percentage of overlapping area – Figure 9 presents the experimental results of *RSGU* for both the (a) syntactic dataset and (b) real dataset with respect to the CPU time when the percentage of overlapping area is varied. In this experiment, we did not compare with the *SGMU* and *VR* algorithms as determining the overlapping area is not considered in these algorithms. Here, in this experiment, the number of users is fixed to 10, the number of group of users is fixed to 16, the number of objects is fixed to 100 and 50000, respectively, while the percentage of overlapping region is varied from 0 to 100 for both datasets. It is obvious that when the percentage of overlapping region increases, the CPU time decreases for both the syntactic and real datasets.



(a)



(b)

Figure9.CPU time with varying percentage of overlapping region over (a) synthetic dataset and (b) real dataset

From these experiments, we can conclude that our proposed framework, *RSGU*, achieved lower CPU time as compared to our previous solution, *SGMU*, and the *VR* algorithm, even when the number of users in a group and the number of groups increase, while the CPU time decreases when the percentage of overlapping region increases. This shows that the fragmentation strategy proposed in our solution, *RSGU*, has significantly reduced the CPU time needed in the computation of subsequent skyline queries of a group of users.

6. CONCLUSION

This paper presents our proposed framework, *RSGU*, which aims at deriving skylines for groups of users by avoiding the unnecessary computation of skylines. Our initial results show that the performance of our proposed framework with respect to CPU time is better compared to [9] and [19]. As future works, we attempt to (1) organise the fragments in a hierarchy [7] so that the scanning time taken to search for the fragments that overlap with the region of a group of users under consideration can be reduced and (2) enhance our framework to identify skylines not only based on the spatial and non-spatial attributes of an object but also the closeness of the object to other interesting objects in the area. This would give more benefit to the users, since they might want to visit a place where there are several other interesting places nearby.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Higher Education Malaysia under the Fundamental Research Grant Scheme (FRGS/1/2016/ICT04/UPM/01/2) and the Universiti Putra Malaysia. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Alwan, A.A., Ibrahim, H., Udzir, N.I., and Sidi, F. Processing skyline queries in incomplete distributed databases, *Journal of Intelligent Information Systems (JIIS)*, 2017, 48(2017): 399-420.
[2] Alwan, A.A., Ibrahim, H., and Udzir, N.I. An efficient approach for processing skyline queries in incomplete multidimensional database, *Arabian Journal for Science and Engineering*, 2016, 41(8): 2927-2943.
[3] Babanejad, G., Ibrahim, H., Udzir, N.I., Sidi, F., and Alwan, A.A. Efficient computation of skyline queries over a dynamic and incomplete database, *Journal of IEEE Access*, 2020, 8(2020):141523–141546.
[4] Bai, M., Xin, J., and Wang, G. Probabilistic reverse skyline query processing over uncertain data stream, *International Conference on Database Systems for Advanced Applications (DASFAA)*, 2012, pp. 17-32.
[5] Bartolini, I., Ciaccia, P., and Patella, M. SaLSa: computing the skyline without scanning the whole sky, *Proceedings of the International Conference on Information and Knowledge Management,* 2006, pp.405-414.
[6] Börzsönyi, S., Kossman, D., and Stocker, K. The skyline operator, *Proceedings of the International Conference on Data Engineering*, 2001, pp.421-430.
[7] Brown, R.A. Building a balanced KD tree in O (kn log n) time, *arXiv preprint arXiv:1410.5420*,2014.
[8] Chomicki, J., Godfrey, P., Gryz, J., and Liang, D. Skyline with presorting: theory and optimizations, *Proceedings of the Intelligent Information Processing and Web Mining*, 2005, pp.595-604.
[9] Dehaki, G.B., Ibrahim, H., Udzir, N.I., Sidi, F., and Alwan, A.A. Framework for processing skyline queries for a group of mobile users, *Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services (iiWAS2018)*, 2018, pp. 331-337.
[10] Dzolkhifli, Z.,Ibrahim, H., Sidi, F., Affendey, L.S., Mohd Rum, S.N., and Alwan, A.A. A skyline query processing approach over interval uncertain data stream with k-means clustering technique, *Proceedings of the Eleventh International Conference on Advances in Databases, Knowledge and Data Applications (DBKDA 2019)*, 2019, pp. 51-56.
[11] Godfrey, P., Shipley, R., and Gryz, J. Maximal vector computation in large data sets, *Proceedings of the International Conference on Very Large Database*, 2005, pp.229-240.
[12] Gulzar, Y., Alwan, A.A., and Turaev, S. Optimizing skyline query processing in incomplete data, *Journal of IEEE Access*, 2019, Vol. 7, pp. 178121–178138.
[13] Halliday, R. Walker. Fundamentals of physics electricity and magnetism,2011.
[14] Khalefa, M.E., Mokbel, M.F., and Levandoski, J.J. Skyline query processing for uncertain data, *Proceedings of the Conference on Information and Knowledge Management*, 2010, pp. 1293–1296.
[15] Kossmann, D., Ramsak, F., and Rost, S. Shooting stars in the sky: an online algorithm for skyline

queries, *Proceedings of the 28th International Conference on Very Large Databases (VLDB'02)*, 2002, pp. 275-286.

[16] Kung, H.T., Luccio, F., and Preparata, F. P. On finding the maxima of a set of vectors, *Journal of the ACM (JACM)*, 1975, 22(4): 469-476.

[17] Lawal, M.M., Ibrahim, H., Mohd Sani, N.F., and Yaakob, R.An indexed non-probability skyline query processing framework for uncertain data, *Proceedings of the 5th International Conference on Advanced Machine Learning Technologies and Applications (AMLTA-2020)*, 2020, pp. 289-302.

[18] Lee, J, Im, H., and You G-W. Optimizing skyline queries over incomplete data, *Information Sciences*, 2016, 361-362: pp. 14-28.

[19] Mohammad, S.A., Ma, G., and Morimoto, Y. A spatial skyline query for a group of users, *JSW*, 2014, 9(11): 2938-2947.

[20] Mohd Saad, N.H., Ibrahim, H., Sidi, F., and Yaakob, R. Skyline probabilities with range query on uncertain dimensions, *Advances in Computer Communication and Computational Sciences, part of the Advances in Intelligent Systems and Computing Book Series (AISC)*, 2018, pp. 225-242.

[21] Mohd Saad, N.H., Ibrahim, H., Sidi, F., and Yaakob, R. Non-Index based skyline analysis on high dimensional data with uncertain dimensions, *Proceedings of the 13th International Baltic Conference on Databases and Information Systems*, 2018, pp. 272-286.

[22] Mohd Saad, N.H., Ibrahim, H., Sidi, F., Yaakob, Y., and Alwan, A.A. Computing range skyline query on uncertain dimensions, *Proceedings of the 27th International Conference on Database and Expert Systems Applications (DEXA 2016)*, 2016, pp. 377-388.

[23] Mohd Saad, N.H., Ibrahim, H., Sidi, F., Yaakob, R., and Alwan, A.A. A framework for evaluating skyline query over uncertain autonomous databases, *Proceedings of the International Conference of Computational Science (ICCS 2014)*, 2014, pp. 1546-1556.

[24] Papadias, D., Tao, Y., Fu, G., and Seeger, B. An optimal and progressive algorithm for skyline queries, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003,pp. 467-47.

[25] Pei, J., Jiang, B., Lin, X., and Yuan, Y. Probabilistic skylines on uncertain data, *Proceedings of the International Conference on Very Large Database*, 2007, pp. 15–26.

[26] Sharifzadeh, M. and Shahabi, C. The spatial skyline queries, *Proceedings of the 32nd International Conference on Very Large Data Bases*, 2006, pp.751-762.

[27] Sharifzadeh, M., Shahabi, C., and Kazemi, L. Processing spatial skyline queries in both vector spaces and spatial network databases, *Journal of ACM Transactions on Database Systems (TODS)*, 2009, 34(3): 14.

[28] Tiger. Available at: http://tiger.census.gov/

[29] Wang, H. and Zhang, W. The τ-skyline for uncertain data, *Proceedings of the 26th. Canadian Conference on Computational Geometry*, 2014.